AFRL-IF-WP-TR-2001-1551

ADAPTERS: A DOMAIN-SPECIFIC PROGRAMMING AND DEVELOPMENT TECHNOLOGY FOR RUN-TIME RECONFIGURATION AT THE SYSTEM LEVEL

JOHN SHACKLETON

Honeywell Laboratories, Inc.
3660 Technology Drive
Minneapolis, MN 55418

JULY 2001

FINAL REPORT FOR PERIOD 25 MAY 1998 – 25 JULY 2001

INFORMATION DIRECTORATE
AIR FORCE RESEARCH LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE OH 45433-7334

[illegible]
KERRY L. HILL
Project Engineer
Embedded Info Sys Engineering Branch
Information Technology Division

Alfred J. Scarpelli
ALFRED J. SCARPELLI
Team Leader
Embedded Info Sys Engineering Branch
Information Technology Division

[illegible]
JAMES S. WILLIAMSON, Chief
Embedded Info Sys Engineering Branch
Information Technology Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 25/07/2001	**2. REPORT TYPE** Final Report	**3. DATES COVERED** *(From - To)* 25 May 98 - 25 July 01

4. TITLE AND SUBTITLE
ADAPTERS: A Domain-Specific Programming and Development Technology for Run-Time Reconfiguration at the System Level

5a. CONTRACT NUMBER F33615-98-C-1320
5b. GRANT NUMBER
5c. PROGRAM ELEMENT NUMBER 69199F

6. AUTHOR(S)
John Shackleton

5d. PROJECT NUMBER ARPI
5e. TASK NUMBER FT
5f. WORK UNIT NUMBER 01

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Honeywell Laboratories, Inc.
3660 Technology Drive
Minneapolis MN 55418

8. PERFORMING ORGANIZATION REPORT NUMBER NA

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Information Directorate
Air Force Research Laboratory
Air Force Materiel Command
WPAFB, OH 45433-7334

10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/IFTA

11. SPONSOR/MONITOR'S REPORT NUMBER(S) AFRL-IF-WP-TR-2001-1551

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release, distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The ADAPTERS project in general is an attempt to fill the current void in system design and integration technology for systems, which include ACS based processing nodes. The approach taken was to develop and integrate ACS tools within existing system level tools. The resulting technology is a more integrated co-design environment, which focuses on three primary technology areas: programming environments, partitioning, mapping, and trade-off analysis, and dynamic run-time environments.

15. SUBJECT TERMS
ACS (Adaptive Computing Systems), reconfigurable computing, field-programmable gate arrays or FPGAs, system design, co-design, programming environments, partitioning, mapping, and trade-off analysis, and dynamic run-time environments

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLEPERSON** Kerry Hill
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	SAR	36	**19b. TELEPHONE NUMBER** *(include area code)* (937)255-6548 x3604

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

Table of Contents

List of Figures

Figure Page

Preface

This work has been supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. F33615-98-C-1320.

Project Contact: John Shackleton (john.j.shackleton@honeywell.com)

1. Project Objectives

The charter for the Adaptive Computing Systems (ACS) project has been to advance the computer technologies that incorporate dynamic hardware configuration capabilities, with the intent of making the resulting devices more prevalent in future DoD systems. Reconfigurable components such as field-programmable gate arrays (FPGAs) have proven to be very promising design options for certain embedded domains. For applications in these domains, ACS-driven systems have reduced size, weight, power, and greatly improved performance and design flexibility compared to more conventional general purpose processor (GPP) and digital signal processor (DSP) systems, without the prohibitive development and maintenance costs of point-solution application specific integrated circuits (ASICs).

Much of the ACS program has focused strictly on the hardware devices themselves, investigating new device subcomponent designs, programming, and compiler technology. In the ADAPTERS[1] project, an ACS thrust led by Honeywell Laboratories, we have concentrated on the challenges of the ACS technologies at the system level. From the beginning of the project, the ADAPTERS partnership of Honeywell Laboratories, University of Southern California/Information Sciences Institute (USC/ISI), and Honeywell Space Systems has recognized that for successful insertion into DoD embedded systems, advances must be made at the system-level design infrastructure.

Adding FPGAs to a GPP and/or DSP creates a heterogeneous system with a significant increase in the system design space. In reality, ACS technology will not operate in isolation, but instead will be part of a mix including reconfigurable devices, GPPs, and DSPs. The performance gains of ACS technology are only realized when the reconfigurable hardware components are integrated into a larger embedded system. The more seamless the integration, the easier it becomes to take advantage of reconfigurable technology. Reconfigurable devices are already difficult to design and program, their run-time adaptability creates an extra design variable previously unexplored in embedded system design, and thus their integration is itself a complex design task.

The benefits of system-level integration improvements are less tangible than improvements at the component level. Processing speedup, for example, is marginal, once system design and integration is complete. However, the schedule and cost savings for the overall system development are obvious. Testing and integration costs are also greatly improved. Tasks that previously required months of development are reduced to days, and seamless integration support gives system designers the tools they need to build ACS solutions in ways previously unavailable.

The overall object of ADAPTERS is therefore to reduce, in terms of programming and development effort, the cost of inserting configurable computing elements into large-scale applications on heterogeneous architectures. The objectives can be further decomposed into the following goals:

[1] ADAPTERS is an acronym that stands for A DomAin-specific Programming and development Technology for run-time Reconfiguration at the System-level

- Reduce ACS programming and integration costs for heterogeneous embedded systems.
- Provide system engineers the means to conquer an increasing system design space as a result of ACS technology.
- Develop design technology that includes run-time reconfiguration requirements at the system level.

The remainder of this document is a summary of the research conducted on the ADAPTERS project. Further details not covered here will be referenced in other ADAPTERS documentation, listed in Table 1.

Table 1: ADAPTERS Technical Documentation

Document	File Name	Description
Missile ATR/Tracking Report	cdrla006-mtrk.doc	Definition of the missile-tracking application and its requirements in the ADAPTERS environment
SAR/ATR Report	cdrla006-sar.doc	Definition of the Synthetic Aperture Radar/Automatic Target Recognition (SAR/ATR) application and its requirements in the ADAPTERS environment
Interfacing Stressmark	cfar-interfacing.doc	Definition of the Constant False Alarm Rate (CFAR) application, defined on the ACS benchmarking project
DADE Users' Manual	cdrla007-users.doc	Users' manual for the Domain-specific Application Development Environment (DADE) system design tool
DADE Template Based Generation of Hardware Specifications	cdrla007-hwgen.doc	Manual describing how the DADE systems design environment models hardware specifications
ATOT-PM Tutorial and Demonstration Notes	cdrla008-atot.doc	Document explaining Architecture Trades and Optimization Toolset (ATOT) and its CFAR demonstration
PML Users' Manual	cdrla008-pml.doc	Users manual for the ADAPTERS specific features of the Performance Modeling Library
SLAAC Tradeoff Study	cdrla009-slaac.doc	Description of the Systems Level Applications of Adaptive Computing (SLAAC) tradeoff study performed on the SLAAC-1 architecture using ATOT
Run-Time Environment Requirements	cdrla010.doc	Run-Time requirements for adaptive computing systems
Dynamic Reconfiguration Run-Time Environment	cdrla011-drrte.doc	Description of the ADAPTERS dynamic reconfiguration run-time environment
Run-Time Reconfiguration Support with Multi-Level Caches	cdrla012-des.doc	Results of the ADAPTERS partial reconfiguration demonstration
Missile-Tracking Demonstration	cdrla012-mtrk.doc[2]	Results of the ADAPTERS missile-tracking demonstration
Space Application Evaluation	cdrla013-space.doc	Results of the Honeywell Space evaluation of the ADAPTERS tool suite
Image Compression Results	image-compression-results.ppt	Results of the image-compression demonstration (viewgraphs)

[2] The *Missile Tracking Demonstration* document has been updated to coincide with the 10/25/2001 release of this final report.

2. Technical Approach

The ADAPTERS project in general is an attempt to fill the current void in system design and integration technology for ACS solutions. At the outset of the ACS program, very little system design support existed for platforms that included reconfigurable devices, and thus our approach has been to develop an integrated co-design environment focusing on three primary technology areas:

1) Programming environments,
2) Partitioning, mapping, and tradeoff analysis techniques,
3) Dynamic run-time environments that support FPGAs.

2.1 Prior State of the Art

Before discussing the ADAPTERS technical approach, first we should briefly describe the state of the art prior to the ACS program inception (around 1997). A number of system-level programming environments have been evolving over the last decade, including the following:

- Ptolemy from UC-Berkeley (http://ptolemy.eecs.berkeley.edu)
- GEDAE from Lockheed Martin Advanced Technology Laboratory (www.gedae.com)
- ObjectGEODE from Telelogic (http://www.telelogic.com/products/objectgeode)
- Simulink from MathWorks (http://www.mathworks.com/products/simulink/)
- Autocoding Toolset from MCCI (http://www.mcci-arl-va.com/initialpage.htm)

Each of these sample toolsets address the obstacles to system level embedded design from a particular angle, although none directly address the issues of ACS design integration. Ptolemy, for example, concentrates on how to take an algorithm and derive design specifications for the configurable hardware devices. The other four samples, GEDAE, ObjectGEODE, Simulink, and the Autocoding Toolset from MCCI, each provide mechanisms for generating system-level application code, but none support automatic integration with reconfigurable platforms. The user of these tools can manually bridge the gap between ACS solutions and more traditional systems implemented on DSPs or GPPs, although such a step requires intimate knowledge of the toolsets and their autocoding nuances.

Similarly, prior to the ACS program, there had been no tool support for system design evaluations and simulation that explicitly targeted reconfigurable hardware platforms. Along with a proliferation of simulation environments developed in the academic community, a couple of the more popular commercial offerings include the following:

- SES/Workbench from HyPerformix (formally a company called SES, http://www.hyperformix.com/products/workbench.htm)
- MATLAB from MathWorks (http://www.mathworks.com/products/matlab/)

Again, with these tools and many like them, the user can take steps to approximate the effects of reconfigurable components on an embedded system, but none have considered reconfigurable issues up front in the design evaluation.

In the area of system run-time environments, a natural evolution toward better ACS support has taken place. ADAPTERS team member USC/ISI has been part of the Systems Level Applications of Adaptive Computing (SLAAC) development effort to standardize board-level application programming interfaces (APIs) and services for ACS applications. Above the API level, however, little prior attention has been paid to other crucial run-time issues:

- Coherent integration with real-time scheduling engines, to manage collections of distributed interdependent reconfigurable tasks
- Communication that is transparent to the application between functions implemented on reconfigurable components and functions hosted on general processing components
- Efficient buffer and memory management in distributed heterogeneous systems that include reconfigurable components (poor memory management increases overhead)
- Middleware service support for partial reconfiguration above the API level.

2.2 ADAPTERS Approach

Our approach is to fill the void in system level design for ACS applications via three components of the ADAPTERS tool suite, each addressing a technology area objective described in first section of this document:

- Domain-specific Application Development Environment (DADE)
- Partitioning and Mapping Tradeoff Environment (PMTE)
- Dynamic Reconfigurable Run-Time Environment (DRRTE)

The ADAPTERS tools suite is summarized here, and the technical details are left to prior ADAPTERS documentation.

2.2.1 Domain-Specific Application Development Environment

DADE is a unified design environment for embedded adaptive systems. A unified environment provides the systems engineer a single graphical notation and tool from which multiple design simulation and code-generation activities may be initiated. Such activities in DADE include the following:

- General modeling of system hardware, application software, and the mapping of software onto hardware
- Auto-analysis of system architecture, end-to-end latency evaluation, and automatic functional partitioning
- Performance modeling and behavioral simulation
- Modeling and automatic generation of hardware specifications for integration with configurable hardware devices.

- Automatic generation of application "glue code" that is compiled and linked into the ADAPTERS dynamic run-time environment; The term "glue code" in this context refers to the software that integrates library function code with a run-time engine.

As shown in Figure 1, DADE is used to model different aspects of an embedded system (e.g., target hardware, system dataflow, functional hardware-software mapping, and mode scheduling) with an emphasis toward configurable computing technology. DADE is also template based, which means that all design components in the unified system model are built from a library of template objects. The template-based approach enables the system engineer to better reuse and parameterize design components. Moreover, the DADE user may select a configuration of software to hardware function mapping from several alternatives, and auto-generate its application code. Such an approach enables the system designer to experiment and test various designs quickly, and without prohibitive debugging of integration code.

The details of DADE and its capabilities are described in the *DADE Users' Manual* and the *DADE Template Based Generation of Hardware Specifications* document.

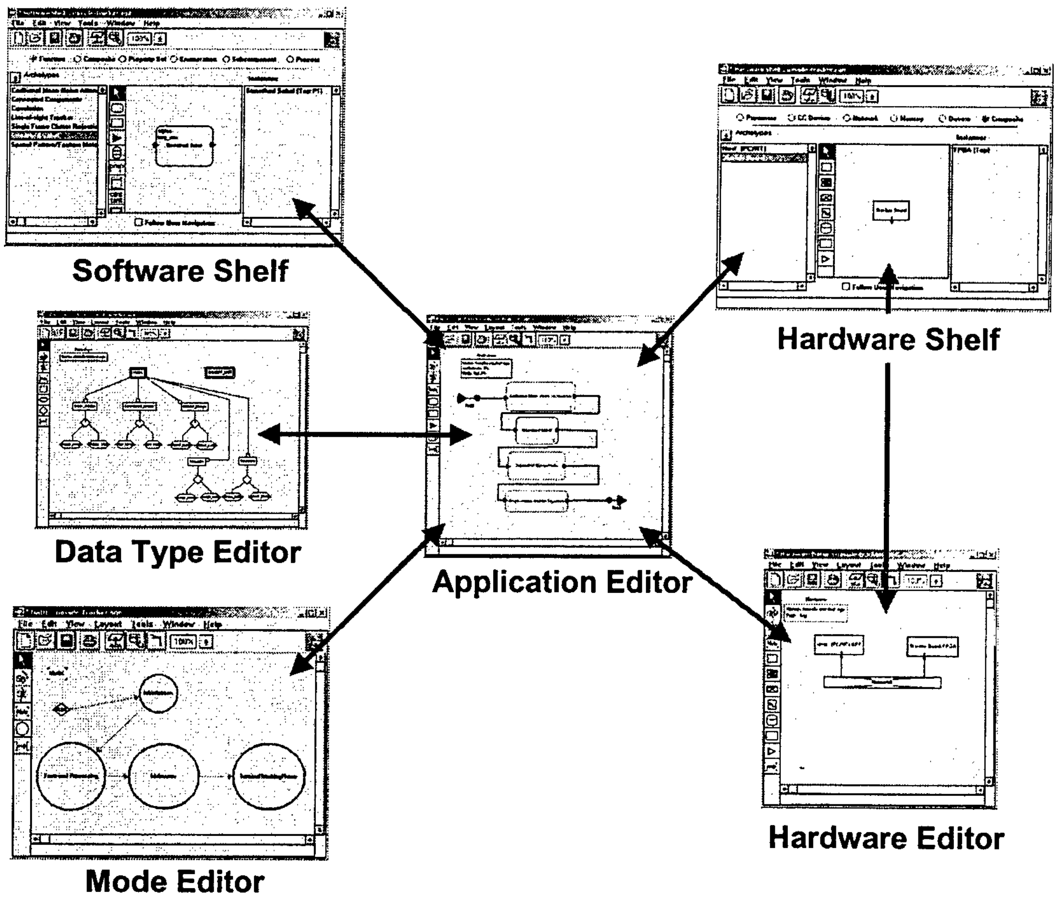

Figure 1: Six DADE Views that Capture Different Aspects of an Embedded System Design

2.2.2 Partitioning and Mapping Tradeoff Environment

The ADAPTERS tools suite leverages two technologies that enable the design engineer to narrow and understand the large complex design space of ACS applications. Collectively these tools comprise the PMTE development thrust of ADAPTERS:

- Architecture Trades and Optimization Toolset (ATOT)
- Performance Modeling Library (PML)

2.2.2.1 Architecture Trades and Optimization Toolset (ATOT)

ATOT is a system analysis tool that performs the following design activities:

- Suggests ways in which functional tasks in the system design are partitioned and parallelized, based on performance requirements
- Searches for a good mapping configuration of function tasks in software-to-hardware processing resources, based on task constraints and hardware capabilities
- Performs an end-to-end latency analysis over candidate software-hardware mapping solutions
- Evaluates the feasibility of implementing certain functions on FPGA devices, and determines the impact on the rest of the system
- Integrates with DADE such that the DADE graphical tool automatically generates ATOT input.

Details on ATOT can be found in the document *ATOT-PM Tutorial and Demonstration Notes.*

2.2.2.2 Performance Modeling Language (PML)

Performance modeling is a special kind of modeling that encourages high-level design abstraction and top-down refinement. Instead of diving into the details right away, performance modeling encourages the designer to concentrate on the most pervasive (and usually most important) system design requirements first. Typically, these requirements focus on general criteria such as component utilization, latency, and throughput. However, in complex systems, it is often necessary to model more detailed criteria, such as intricate hardware/software interactions, operating system overheads, cache and memory access, and network protocol behavior.

The PML addresses these issues. The PML is a prototyping library based on the Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL) that aids the design of system level models. It has the following features:

- A robust processor model that captures the high-level performance behavior incorporated in most commercially available processors, including on-board memory and cache access, interrupt handling services, and instruction set characterizations.
- Software modeling that operates separately from the hardware characterizations, and encourages hardware/software co-design.

- Task scheduling capabilities that implement local and remote task communications, preemptive tasking, static or dynamic scheduling, rate monotonic scheduling, distributed scheduling, and other services one would expect from a run-time executive or a general purpose operating system.
- A lightweight processor model that allows the designer to create single-threaded applications without the overhead of a complete operating system.
- Report generators that automatically provide detailed accounts of processor task activities, missed deadlines, processor utilization, task utilization, and overall latency.

Moreover, the PML has been extended for the ADAPTERS effort to include an FPGA component that models the behavior, overhead, and processing capabilities of a typical adaptive computing device. This FPGA component is fully parameterizeable, and is designed to support the general performance modeling requirements of all adaptive computing systems. Further details on the PML are found in the document *PML Users' Manual.*

2.2.3 Dynamic Reconfigurable Run-Time Environment

Adaptive computing uses reconfigurable elements in combination with other computing resources to achieve speedup for compute-intensive algorithms. While adaptive computing has shown promise in providing high performance, the low level at which it is programmed makes software support for adaptive computing critical to its success. Software support can be divided into two types: design tools and run-time support. Design tools such as DADE are used to help the system designer map applications to the programmable logic and other parts of the adaptive computing system. However, design tools alone are not sufficient to take advantage of the computing power of adaptive computing systems. A run-time environment is needed to support application programming in much the same way an operating system is needed in a general purpose computing system. The run-time support abstracts the system designer from having to deal with low-level programming details, such as which control lines need to be manipulated to communicate data from one subsystem to another or how to configure an FPGA.

The ADAPTERS team has brought three facets together to address the area of dynamic reconfigurable run-time environments:

- SLAAC board APIs and services
- MetaH, mode based scheduling kernel
- Dataflow based scheduling run-time associated with DADE.

2.2.3.1 SLAAC (Systems Level Applications of Adaptive Computing)

SLAAC creates an open, standards-based, scalable systems definition and a commercial off the shelf (COTS) based reference-platform implementation distributed to the ACS research community to establish interoperability, lower the barriers, and accelerate the integration of new research results into defense testbeds and flights. Particular to the ADAPTERS effort are the SLAAC board-level APIs utilized in the application demonstrations, and the use of the SLAAC-1 and SLAAC-1V boards. Moreover, an innovative multilevel caching scheme has been developed on ADAPTERS to improve performance of applications that are capable of partial reconfiguration, the details of which are found in the document *Run-Time Reconfiguration Support with Multi-Level Caches.*

2.2.3.2 MetaH

MetaH is an architecture description language (ADL) developed originally at Honeywell Laboratories from which several design activities may take place:

- A closed form system is modeled in terms of hardware components, software processes, system modes, and the modal transitions
- The system is formally validated and its schedulability analyzed
- System requirements are automatically generated into a real-time application kernel.

The ADAPTERS project builds upon MetaH to demonstrate mode-scheduled applications in a configurable computing environment. Details on the insertion of MetaH into the ADAPTERS demonstrations are covered in the document *Dynamic Reconfiguration Run-Time Environment.*

2.2.3.3 DADE Run-Time

The ADAPTERS dataflow based execution environment provides the dynamic, event-triggered run-time support for most of the ADAPTERS demonstrations. In summary, the run-time capabilities include the following:

- Seamless integration with the DADE system design environment
- Hosting of an application across a distributed heterogeneous platform
- Automatic management of remote communications and buffers
- Automatic management of parallel tasks, data distribution, and multitasking
- Automatic management of ACS board-specific requirements (currently supporting the WILDFORCE, STARFIRE, and SLAAC-1 board implementations).

Details on the DADE dataflow run-time are also covered in the document *Dynamic Reconfiguration Run-Time Environment.*

3. Summary of Accomplishments

The following section summarizes the accomplishments of the ADAPTERS project, categorized by its three main technology areas: programming environments, partitioning and mapping tradeoffs, and dynamic run-time environments.

3.1 Programming Environments

Throughout the ADAPTERS project, DADE serves as the launching point for all other development activities. DADE models have been built to capture the software, hardware, and system mapping requirements of all the ADAPTERS demonstrations. In addition, DADE was used to define the hardware specifications for a sample CORDIC application implemented on the Annapolis Microsystems WILDFORCE board, documented in *DADE Template Based Generation of Hardware Specifications*. Thus DADE acts as the central repository for all ADATPERS architecture and integration designs.

From our work with DADE, a general methodology has emerged. The following lists the general process we came to use when employing DADE on an application:

1. First import libraries of existing application functions into DADE
2. For functions not modeled in an existing DADE library, function archetypes are created on the DADE shelf repository, which identifies the platforms on which the function is implemented, and its associated parameter list
3. The data types required by the functional flow of the system are modeled in the Data Type Editor
4. Functions are then copied from the shelf repository to the Application Editor, to describe the functional data-flow of the application
5. Data types are assigned to the functional flow in the Application Editor, thereby defining the types and sizes of data passed through the functions
6. For functions implemented on reconfigurable devices, such as an FPGA, the user has the option of defining its hardware specification; the specification may be a complete design that is used to create an image for the device, or it may simply act as integration code for existing hardware specification components
7. At some point in the process, the system hardware architecture is modeled in the Hardware Editor, the boards and processing devices in the system, the buses and memories
8. Functions from the Application Editor are then mapped to computing resources in the Hardware Editor; platform-specific information about the functions, such as their parameters lists, are inferred from the shelf repository
9. Optionally, the user may group the functions into modes in the Mode Editor
10. Once the application is modeled, the user can initiate other design activities: automatically generate hardware specifications, pass design information to ATOT for further evaluation, or automatically generate glue code that is linked and compiled with the DADE run-time for execution on the target platforms.

It is important to note what DADE does not attempt to do as a design tool. DADE does not design algorithms for implementation on ACS platforms, nor does it generate the application code itself. Other tools exist, such as MATLAB, which are used for application and algorithm development. Instead DADE generates the surrounding glue that integrates existing modularized application code into the DADE run-time environment. The glue-code handles local and remote function-to-function communication and data distribution. Also, DADE is not meant to design hardware components. Although it does model hardware at the system architectural level as a reference to application functions, DADE is not the right tool if you need to design the hardware devices themselves. Overall, DADE is a system design and integration technology employed to rapidly connect together libraries of functions to be hosted on distributed heterogeneous platforms that include reconfigurable devices.

Honeywell Space Systems has done an evaluation of DADE that is found in the document *ADAPTERS Space Application Evaluation.*

3.2 Partitioning and Mapping Tradeoffs

Both ATOT and PML, the two technologies applied to this problem area, were developed prior to the ADAPTERS project. ATOT had been an ongoing research activity for years, developed internally at Honeywell Laboratories. The PML has its roots in the Defense Advanced Research Projects Agency (DARPA) Rapid prototyping of Application-Specific Signal Processors (RASSP) program that completed in 1997. Each technology was extended on the ADAPTERS project to address the special needs of ACS systems.

ATOT was extended to model the effects of reconfigurable devices on an embedded heterogeneous system. Characterizations of an FPGA device were added to ATOT, so that the system designer can more accurately gauge the design effects of the FPGA components in the system. Measurable design criteria to include end-to-end latency, throughput, and the effects of parallelism. The updated ATOT toolset also makes suggestions on how functions can fit on an FPGA based on their physical layout requirements. A first-order analysis can be made before expensive place-and-route designs are performed, thereby saving valuable design time. USC/ISI implemented a sample CFAR application on the SLAAC platform and then used ATOT to perform a tradeoff analysis. Their evaluation is found in the document *SLAAC Tradeoff Study.*

The PML was extended in ADAPTERS to include in its modeling library an FPGA device. The modeling component includes a number of behavioral characteristics, such as the functional modules on the device, as well as physical characteristics. The FPGA modeling component interoperates with the other PML components in its token-based framework. An example of its use is defined in the document *PML Users' Manual.*

3.3 Dynamic Run-Time Environments

When we integrate functions at the system level, eventually we must determine how the functions are scheduled for execution. In an ACS system, our question takes on added meaning because function(s) implemented on reconfigurable devices have additional overhead when new configurations are loaded. For the class of applications addressed by the ACS program, we have two primary paradigms for run-time scheduling: dataflow scheduling and mode-based scheduling. For ADAPTERS, we first addressed the two approaches separately in their own demonstrations and then brought them together for the final demonstration of the project.

MetaH was first applied to a sample image compression demonstration, implemented on an Annapolis Microsystems' STARFIRE board hosted by Pentium/NT platform. MetaH is a natural fit for ACS embedded applications because device reconfigurations most often occur as a result of a mode or state transition in the overall system. For example, as in the image compression demonstration, when a new stage in the compression algorithm is encountered, a mode change is initiated, and the underlying FPGA requires a new configuration download. Results of the image compression demonstration are documented in the *Image Compression Results* viewgraphs. Figure 2 illustrates its mode architecture.

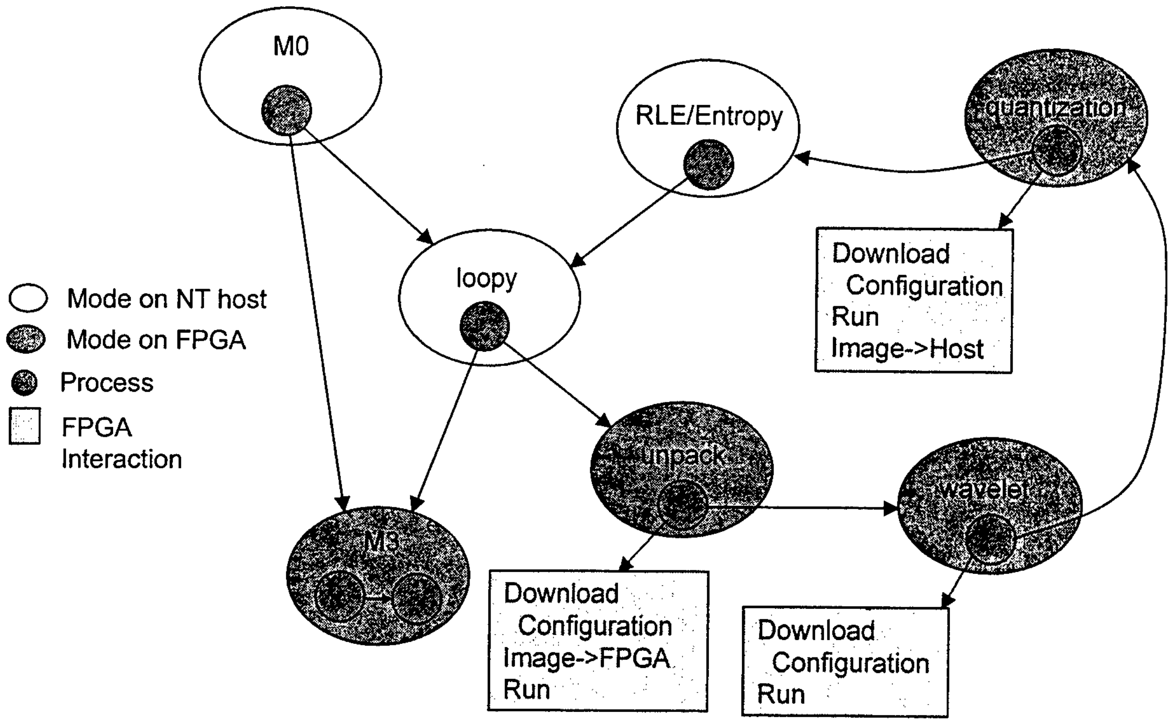
M0
RLE/Entropy
quantization
loopy
Mode on NT host
Mode on FPGA
Process
FPGA Interaction
unpack
wavelet
M3
Download Configuration Run Image->Host
Download Configuration Image->FPGA Run
Download Configuration Run

Figure 2: The Mode Interaction of the Image Compression ADAPTERS Demonstration

The DADE dataflow run-time has been applied to the CFAR and missile-tracking demonstrations. Both demonstrations are described in detail in the document *Dynamic Reconfiguration Run-Time Environment.* Further detail of the missile-tracking demonstrations and its results are described in the document *ADAPTERS Missile Tracking Demonstration.* In these demonstrations, functions were connected as a series of pipelined computations. In this environment, a function is triggered when it has data ready in its incoming data buffer. The CFAR example was implemented on an Annapolis Microsystems WILDFORCE board with a Pentium host and on the SLAAC-1 board with a Pentium/NT host (Figure 3). The missile-tracking application has been implemented on the several heterogeneous platforms, including the WILDFORCE platform, an Annapolis Microsystems' WILDSTAR board hosted by a PowerPC board running VxWorks, and on an Annapolis Microsystems' STARFIRE board hosted in a Pentium/NT platform (Figure 4).

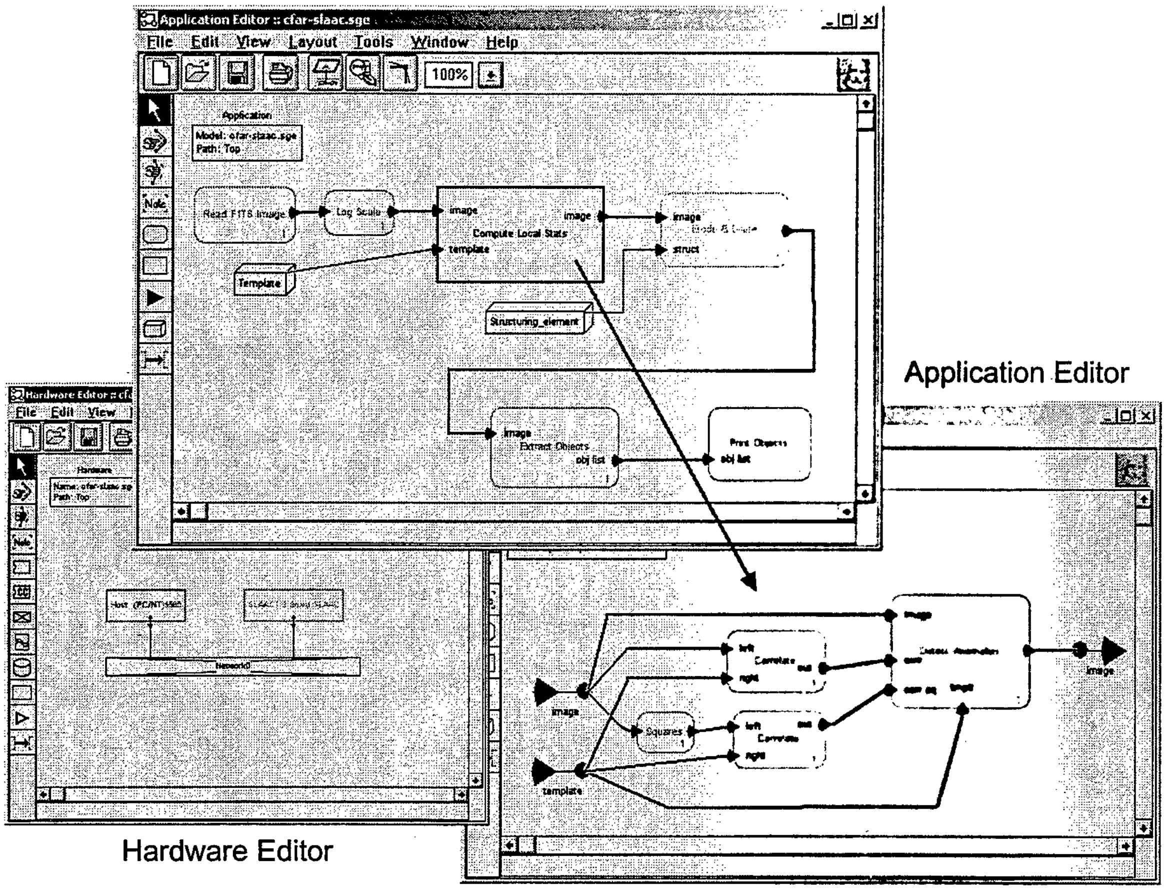

Figure 3: A DADE Capture of the CFAR Demonstration Implemented on the SLAAC-1 Platform

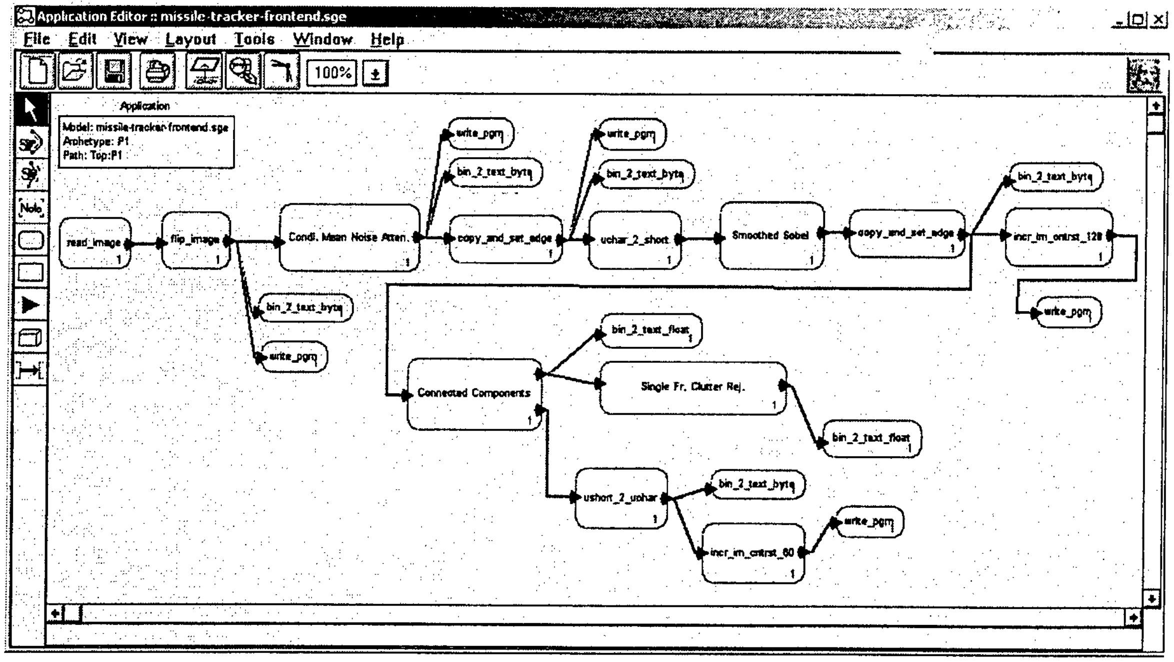

Figure 4: A DADE Capture of the Functional Dataflow of the Missile-Tracking Demonstration

The final ADAPTERS demonstration is a sample implementation of the data encryption standard (DES). Our DES demonstration is implemented on a SLAAC-1V board, and addresses two important technical issues of ADAPTERS. First and foremost is partial reconfiguration, the ability of an application to partially reconfigure a device so that overhead is reduced while still maintaining the application dataflow. From the beginning of reconfigurable technology, it has been the hope that partial reconfiguration will become viable and help make dynamic adaptive hardware components even more attractive design alternatives. Our DES example is a step toward that end (partial reconfiguration is addressed further in the conclusions section of this document).

The multilevel caching implemented in the DES demonstration improved reconfiguration overhead by 40 times. Results are documented in *Run-Time Reconfiguration Support with Multi-Level Caches*.

The second issue addressed by the DES demonstration is the combination of mode-based and dataflow-based scheduling paradigms within the same run-time environment (Figure 5). The mode-based scheduling approach of MetaH was incorporated in the dataflow scheduling software infrastructure of the DADE run-time, such that both scheduling options are available within the same dynamic run-time. When both paradigms are active in an application at the same time, the mode determines which dataflow pipeline is active. An active set of functions will maintain the current dataflow until another mode change occurs. At the present time, the combined mode/dataflow scheduling is limited to applications that are hosted on a single board. The infrastructure has not been put in place yet to support mode transitions that reach across distributed nodes (and this general research probably falls outside the scope of ACS).

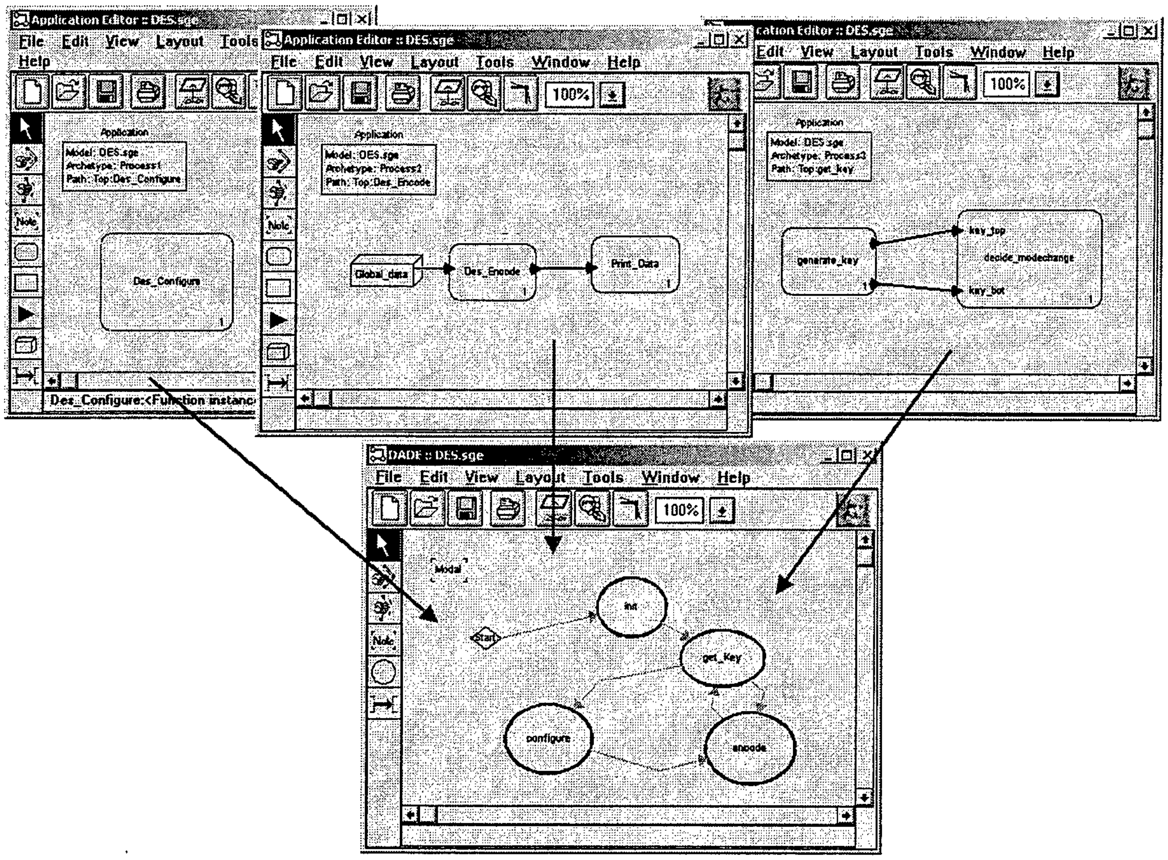

Figure 5: A DADE Capture of the DES Demonstration Combining Mode-Based Scheduling with Dataflow Based Scheduling

3.4 Technology Transfer

Honeywell Space Systems was added to the ADAPTERS project to provide a perspective that is grounded in real-world applications. Their role on the project has been twofold: to use their experience to evaluate the ADAPTERS toolset, and to identify an application that would further exercise the tools. Their results are documented in the *Space Application Evaluation*.

In addition, as the ADAPTERS project has proceeded, a commercial version of DADE and the DADE run-time has emerged through an independent development path initiated by a group at Honeywell Space Systems[3]. The commercial version of DADE is called SAGE (Systems and Applications Genesis Environment) and includes a number of system design features inside and outside the scope of ACS. Information on SAGE is located at the web site www.honeywell.com/sage. Features in the commercial version of the toolset that are a direct result of the ADAPTERS development include:

[3] The commercial tools development group is a different group than those who performed the ADAPTERS evaluation, although the two groups work together and are located in the same facility.

- Hybrid mode-based and dataflow based scheduling in the run-time environment (the idea originated from the ADAPTERS image-compression demonstration)
- The Mode Editor in DADE
- Most of the template-based programming features of the DADE shelf repository
- The automatic VHDL specifications generated by DADE
- Support for reconfigurable hardware devices in the run-time.

4. Conclusions and Future Work

Research like ADAPTERS is important because it recognizes that the largest costs in embedded systems design and maintenance are no longer at the component level, but at the system level where integration takes place. Current trends in military systems are toward COTS solutions as much as possible, and as a result, more complexity is pushed into the layers of the system that service COTS components and bridge the communications between them. Trends are also toward hosting more functionality onto fewer computing devices, or reducing the number of boxes that an embedded military application requires. Consequently, the integration task becomes even more complex, because computing resources must be shared among a growing number of application tasks.

ACS technology fits well into this vision. Reconfigurable devices such as FPGAs require less power, are often smaller than GPP and DSP devices, and offer better performance. While FPGAs in general are not as fast as hard-wired ASICs, their hardware is less expensive to develop because of their COTS availability, and random access memory (RAM) reconfigurable FPGAs are dynamically programmable and therefore much more maintainable than ASICs.

FPGAs should not be used everywhere. GPPs and DSPs are more cost-effective solutions for many applications, and ASICs still have a niche in embedded system design. For those applications that do fit the criteria for FPGA implementation, however, such as single instruction, multiple data (SIMD) or multiple instruction, multiple data (MIMD) compute-intensive pipelined applications, ACS technology is the way to proceed, assuming enough system design and run-time support is available.

Our concluding observations will be presented in the following four categories:

- System design tool support
- Run-time integration support
- Partial reconfiguration
- Future ACS domains.

4.1 System Design Tool Support

A gap currently exists in design tool technology between a system level of abstraction and design at the component level. Computer aided design (CAD) tools like Synplicity enable a design engineer to build an FPGA component, but very few features are in the tools that enforce the system level requirements that will impact component design. Conversely, high-level design tools have very little visibility into the low-level design of components, and thus system level architectures are often over designed with too much redundancy. It is a constant tradeoff, between hiding detailed component-level design information, and making this information available to improve component integration of a design at the system level. Design engineers are forced to understand all the requirements that overlap system level and component level designs without much current tool support.

Design tools at the different levels of abstraction in general do not speak the same language; they require different types of domain knowledge, and they often work from a conflicting set of goals. For example, at the component level, the design may try to optimize performance, while at the system level the goal is future maintainability, and as experience shows, these requirements often work against each other. This dichotomy is particularly obvious in ACS applications, in which configurable components by their nature require low-level hand tweaked detail to improve performance, thereby making their seamless integration into a larger heterogeneous system more difficult.

The ADAPTERS works shows that such a gap in tool technology exists, and it is not a clear problem to solve for embedded heterogeneous systems. The final answer more likely calls for a more varied tailorable approach. Instead of trying to find a huge monolithic tool suite that satisfies every design activity, the answer more likely lies in a collection of loosely coupled tools that provide more flexible, programmatic properties and interfaces. Design tools of the future should become more plug-and-play so that design information is more easily shared between levels of abstraction and between different applications. In the design environments of the future, especially in the context of ACS technology, one size no longer fits all.

The other issue that ADPATERS uncovered in terms of design tools is the need for system level integration support for ACS applications. Reconfigurable devices will not operate in a vacuum, but will reside within heterogeneous systems sharing computational responsibilities with other kinds of components. As a result, the design tools used in the future will have to model, evaluate, and generate design artifacts for systems that include reconfigurable devices, as well as more traditional components such as GPPs and standard buses. This is the approach we started under ADAPTERS. A few general rules have emerged:

- Unlike general software functions that have no physical limitations to their mapping, functions implemented on an FPGA are constrained by their physical layout and I/O. Not every function implemented on an FPGA can be co-located with another function on the same device, and the mapping of functions to reconfigurable devices is a non-deterministic polynomial (NP) complete problem.
- The designers of FPGA components and the designers of system-level architectures will in reality come from different engineering backgrounds, and likely will not speak the same language. They will, however, have to learn to work together since many of their requirements will overlap. Integrated tool technology can help overcome these barriers.
- It is best to perform system analysis and tradoffs as soon as possible in the design process. The sooner the impact of certain low-level design decisions is understood at the system level, the more efficient the system architecture will become, and the easier it will be to address major design integration problems.

4.2 Run-Time Integration Support

Thanks to the SLAAC effort, the ACS community is moving toward standardization on how to manage adaptive applications at the board interface level. Nevertheless, the ramifications of this movement will impact how applications are developed at the system level, and we must continue to gain a better understanding of these implications. Improved understanding of integrated systems will happen naturally over time, as the board-level APIs and services are further refined. The wider the scope of applications that are implemented with a COTS-based interface, the more applicable and robust the interfaces and APIs become.

Over the last decade, we have seen embedded operating systems increase their support services as well understood application tasks are pushed down into the operating system layer of software. A similar fate is destined for run-time software that supports ACS platforms. As the community builds up domain knowledge of what kinds of services are most useful, those services will gradually migrate from the application layer to the run-time support layer. The same is true for the lower levels of abstraction as well. As COTS hardware vendors learn what configurations and behaviors are most important to the component developers, then such support as programmable macros and configuration libraries will trickle up to the board-level API and become more accessible. The key is that the hardware vendors must accept the trend toward API and service standardization.

Also, ACS run-time support in the future must facilitate design integration between reconfigurable devices and other components in the system. The ADAPTERS project has been a first step toward this goal. Without a run-time layer that enables seamless integration between heterogeneous components, system engineers will spend too much time building the connections (i.e., buffer management, queue maintenance, timing mechanisms) by hand. Such development is error prone, specific to the immediate problem, and is not easily reused between different applications and design activities, wasting time that is better spent evaluating and improving system architectures.

ACS run-time support of the future will include task scheduling capabilities. As shown in the ADAPTERS demonstrations, run-time scheduling is needed to manage applications at the system level and ensure smooth transitions between reconfiguration requests with as little overhead as possible. Many scheduling techniques have been developed for embedded systems, and it would be worthwhile for someone to evaluate which of these techniques can also be applied to the specialized requirements of ACS applications. On the ADAPTERS project we have applied the mode-based scheduling of MetaH, and the dataflow (event-triggered) scheduling of the DADE run-time execution environment.

The next generation of ACS run-time support will have more intelligence. A current trend in general middleware support is to make the run-time layer reflective, meaning that the run-time support is aware of the special dynamic requirements of the application execution, and can adapt the application accordingly. For example, as a particular ACS application is running, the run-time support may recognize that the algorithm it is currently using is suboptimal, based on the trends in the data, and a reconfiguration to a different algorithm implementation would improve performance. Such an approach creates a two-tiered support infrastructure for adaptation, coordinated between the system level and the component level. Other benefits include the following:

- Flexible control of multiple functions on shared reconfigurable devices
- Intelligent control of reconfiguration downloads to reduce software overheads
- Applications that better adapt to the state conditions (algorithm adaptation)
- Improved fault tolerance.

The ACS project has identified the value of reconfigurable hardware platforms. For ACS to reach its full potential in embedded military applications, the software run-time support that drives the hardware will have to evolve and take advantage of adaptation more intelligently.

4.3 Partial Reconfiguration

In the ADAPTERS project, we took a first look into the applicability of partial reconfiguration. The results have been a mixed bag of successes (highlighted by our DES demonstration), along with some observations and unresolved issues:

- Partial reconfiguration has potential for a specialized class of applications, where execution parameters impact a modularized portion of an algorithm implementation.
- Partial reconfiguration is difficult to implement, and must be considered on a case-by-case basis because of physical layout constraints.
- Widespread use of fine-grained partial reconfiguration is unlikely, because its implementation is too difficult to generalize for a larger set of applications. More research is required to determine the characteristics of partial reconfiguration with respect to single-FPGA vs. multi-FPGA solutions, and the relationships between the method used to reconfigure the FPGA vs. the improvements in application performance
- If partial reconfiguration is to succeed, board-level APIs and services must continue to evolve. The multilevel caching technique developed by USC/ISI for the ADAPTERS DES demonstration is a good example.
- Partial reconfiguration requires better tool support, to gain visibility into the effects of partial reconfiguration and understand its limitations. Current design technology is limited to the static domain, and does not currently address real-time dynamic issues.
- New candidate applications should be explored to identify where partial configuration works the best.

4.4 Future ACS Domains

A key result of the ADAPTERS project is the affirmation that ACS technology has a place in future space platforms. Consider the following observations:

- A trend towards diminishing COTS support is emerging for military/space (and even avionics) applications, especially in the areas of operation in extreme environments, and fault-tolerance. FPGA vendors typically do offer both avionics and military grade products, making life-extension programs possible for certain systems.

- Future requirements of space-based applications fit ACS technology well. More functionality will be hosted on a smaller number of compute devices, and reconfigurable technology provides the most flexible approach.
- FPGAs take up the same amount of physical space as GPPs, but use less power and deliver higher performance.
- An FPGA solution has a built-in fault tolerance.
- Most space applications operate with real-time constraints that fit the right granularity for ACS adaptation. Mode changes trigger reconfiguration in the system, but they are relatively infrequent, and adaptation occurs on the order of seconds, which puts the reconfiguration overhead within a tolerable range.
- Candidate applications include high-throughput payload processing applications: hyper-spectral imaging, space-based radar, RF surveillance, and digital on-board radio.

Another future domain for ACS technologies lies at the other end of the spectrum. While space platforms represent compute-intensive high-tech domains, we should not neglect simpler problems that can benefit from reconfigurable computing. In the world today there are many simpler applications that use ASIC technology that could be made more useful with an FPGA upgrade. For example, wireless phones can potentially host more functionality through RAM-reconfigurable technology without increasing the size of the device. Downloading a new configuration, instead of replacing the entire physical unit can make hand-held device upgrades more attractive. Hundreds of potential applications are out there, and should not be limited to large-scale domains.

5. Academic Papers

S. Kumar, et al., “ADAPTERS,” *Proceedings of the MAPLD ’99 Conference*, Johns Hopkins University, Applied Physics Laboratory, Laurel, Maryland, September 28-30, 1999.

S. Kumar, et al., “Programming and Development Environments for Configurable Computing Systems”, which appeared in the *Proceedings of the IEEE Aerospace 2000 Conference*, Big Sky, Montana, March 18-25, 2000.

6. Acknowledgements

Many people worked on the ADAPTERS project to make it a success:

Honeywell Laboratories: Devesh Bhatt, Rashmi Bhatt, Kevin Driscoll, John Fischer, John Golusky, Bala Kannikeswaran, Sanjaya Kumar (the original principal investigator of the project), John Shackleton, Hazel Shirley, Steve Vestal, Mark Vojta, Bill Wren, and Phil Zumsteg

USC/ISI-East: Peter Bellows, Stephen Crago, Dong-In Kang, Bob Parker, and Brian Schott

Honeywell Space Systems: Gary Gardner and Jeremy Ramos

LIST OF ACRONYMS

ACRONYM	DESCRIPTION
ACS	Adaptive Computing Systems
ADAPTERS	A DomAin-specific Programming and development TEchnology for run-time Reconfiguration at the System-level
ADL	Architecture description language
API	Application programming interface
ASIC	Application-specific integrated circuit
ATOT	Architecture TradeOff Toolset
ATR	Automatic target recognition
CAD	Computer aided design
CFAR	Constant false alarm rate
COTS	Commercial off the shelf
DADE	Domain-specific application development environment
DARPA	Defense Advanced Research Projects Agency
DES	Data encryption standard
DRRTE	Dynamic reconfigurable run-time environment
DSP	Digital signal processor
FPGA	Field programmable gate array
GPP	General purpose processor
I/O	Input/Output
MIMD	Multiple instruction multiple data
NP	Non-deterministic polynomial
PML	Performance modeling library
PMTE	Partitioning and mapping tradeoff environment
RAM	Random access memory
RASSP	Rapid prototyping of application-specific signal processors
SAGE	Systems and application genesis environment
SAR	Synthetic aperture radar
SIMD	Single instruction multiple data
SLAAC	Systems Level Applications of Adaptive Computing
USC/ISI	University of Southern California/Information Sciences Institute

VHDL	VHSIC hardware description language
VHSIC	Very high speed integrated circuit